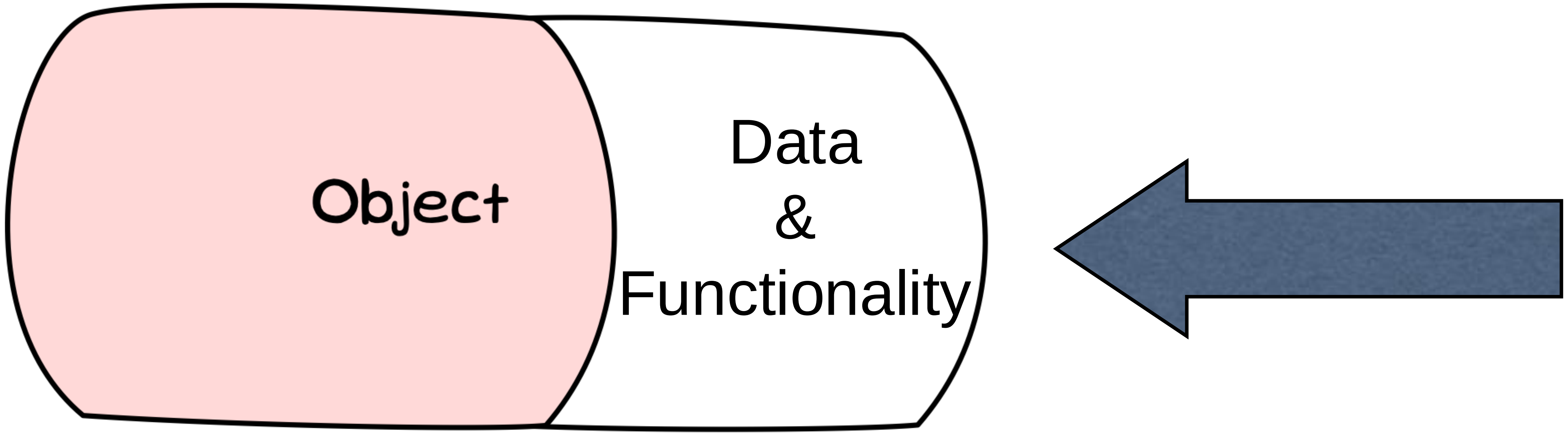
Inheritance

Dr. Viet Vo
vvo@swin.edu.au
Department of Computing Technologies
School of Science, Computing and Engineering Technologies

Learning Outcomes

- The importance of generalisation and specialisation in OOP
- Understand how to implement inheritance
- Demonstrate inheritance with real-world examples

Object oriented programs contain objects that know and can do things



Object

Data
&
Functionality

Know things
&
Do things

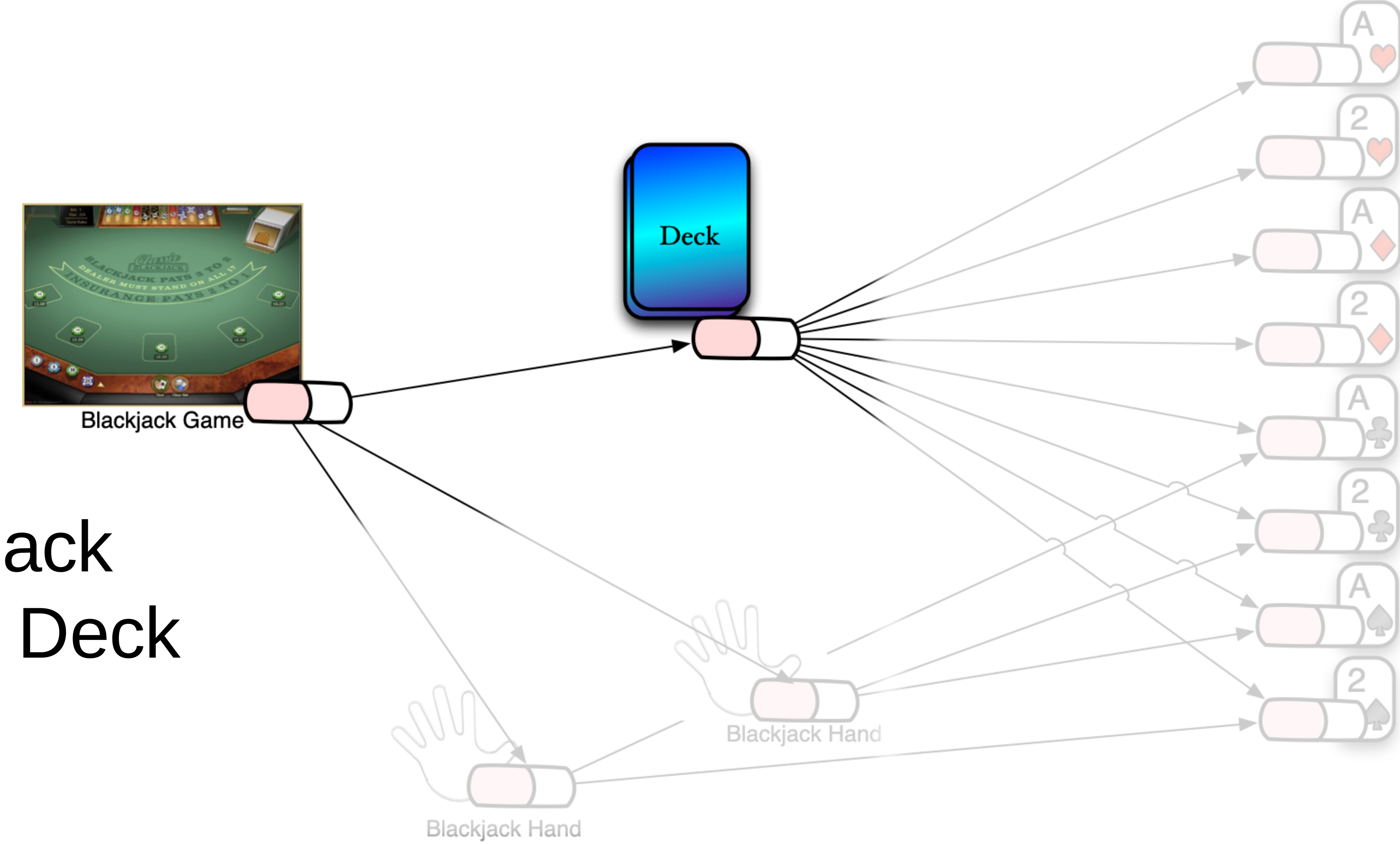
Remember there are three main kinds of relationships

Association

Aggregation

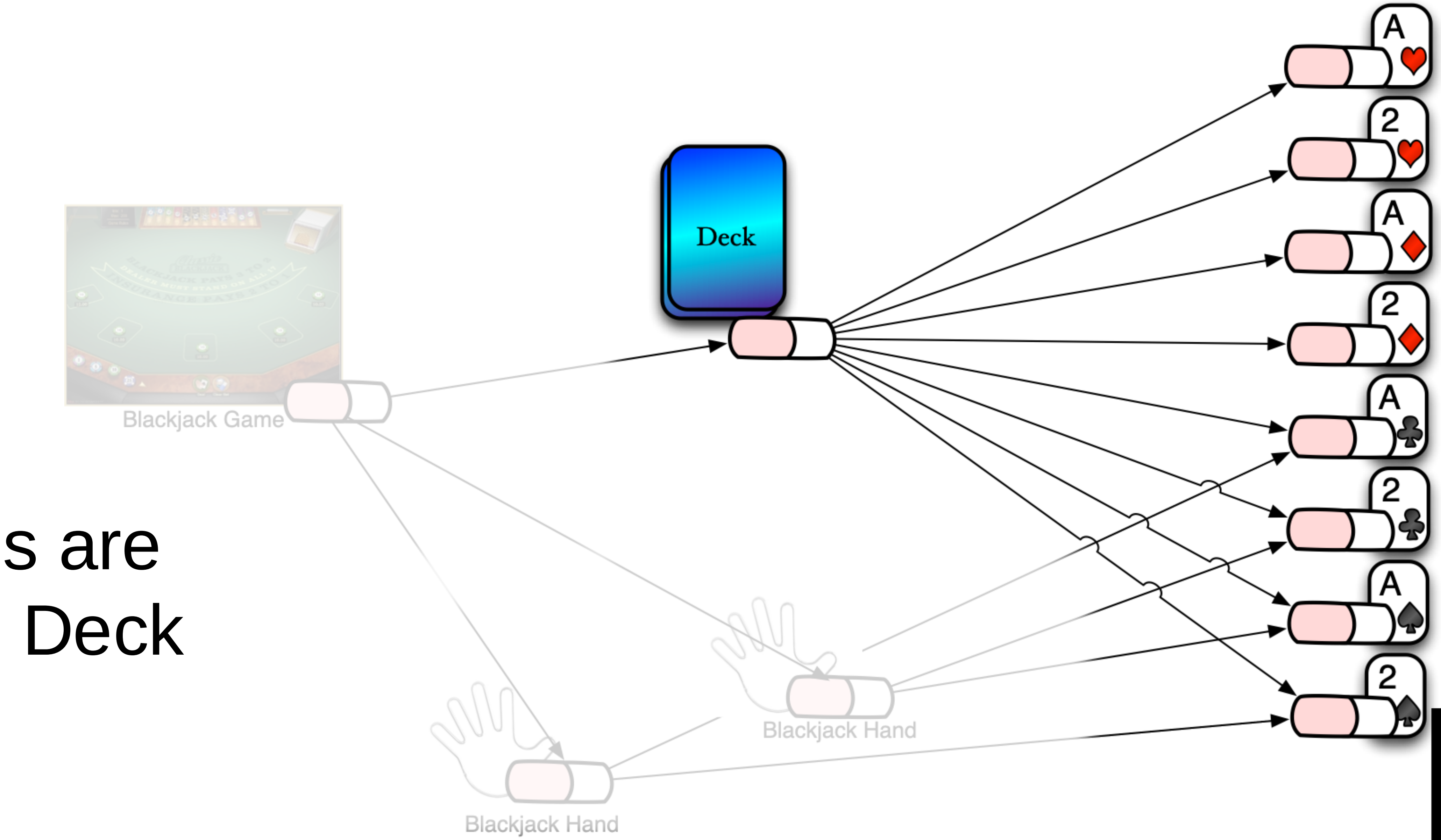
Dependence

Use association for "has-a" kind relationship



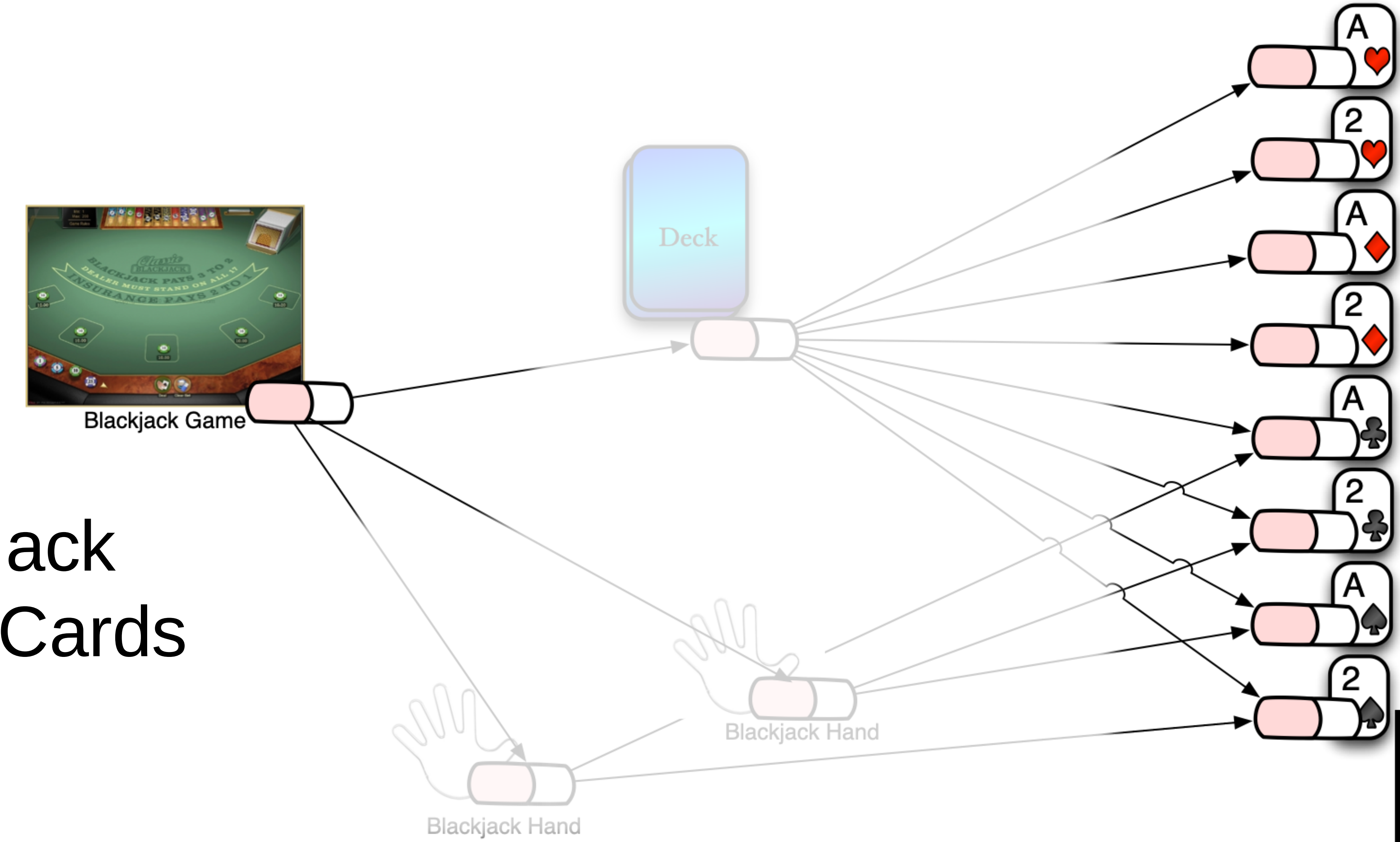
The Blackjack Game **has a** Deck

Use "aggregation" for whole-part or container relationships



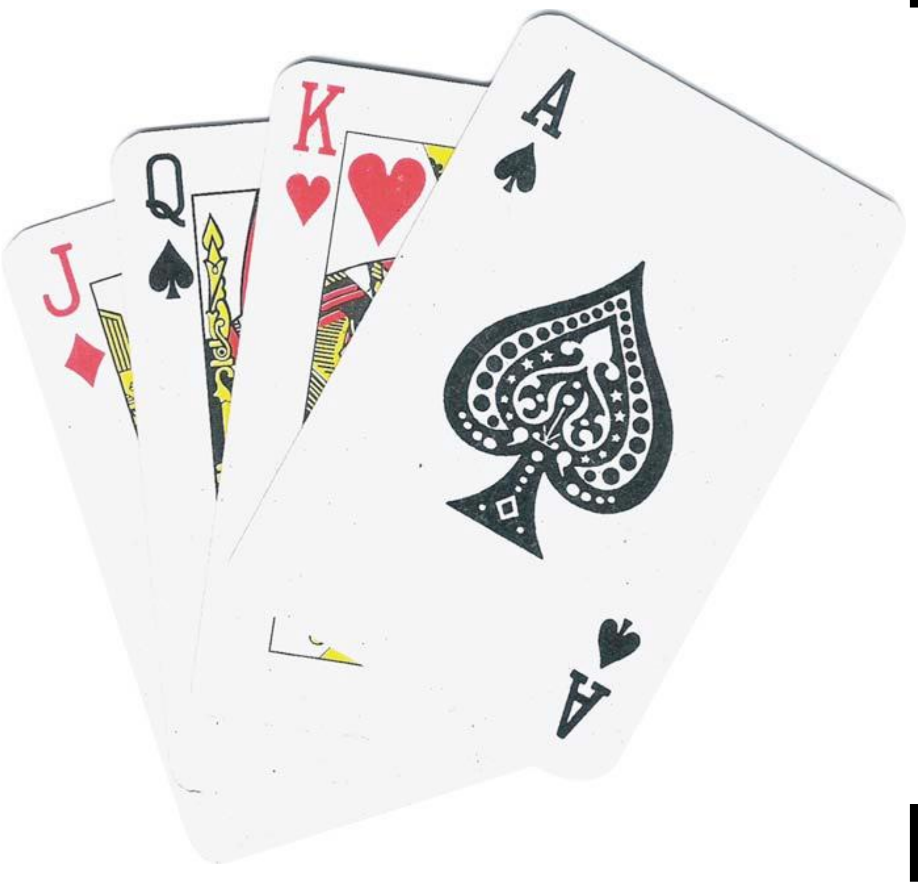
The Cards are part of the Deck

Use dependency for temporary "uses" style relationships

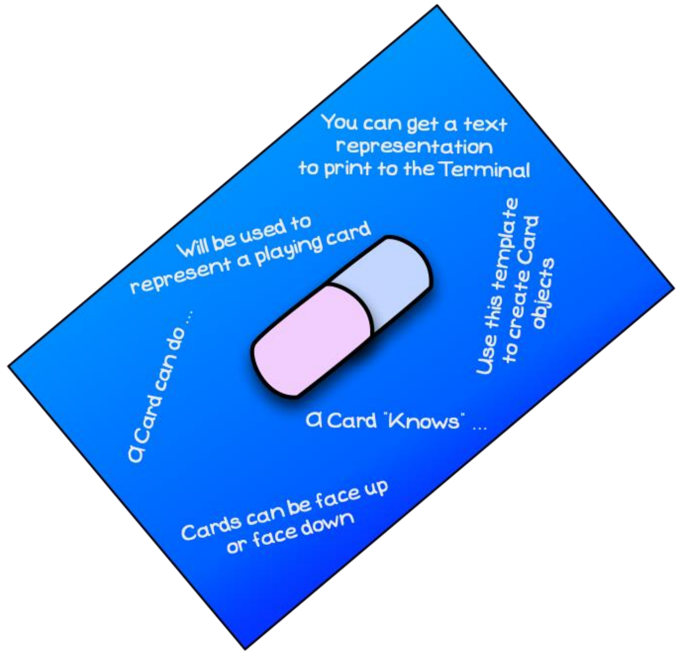


The Blackjack Game **uses** Cards

Developers use the process of abstraction to define object classes



Specification for a Card



You can get a text representation to print to the Terminal

Will be used to represent a playing card

Use this template to create Card objects

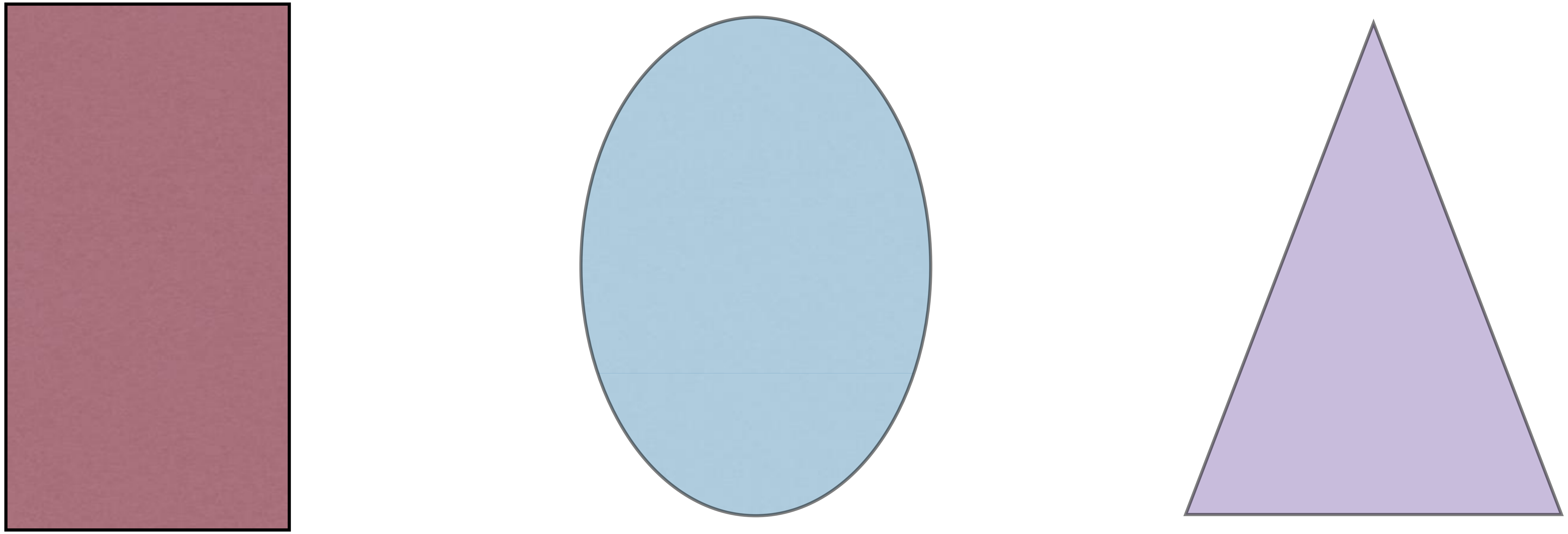
a Card can do ...

a Card "Knows" ...

Cards can be face up or face down

Classification

Abstraction also includes generalisation and specialisation

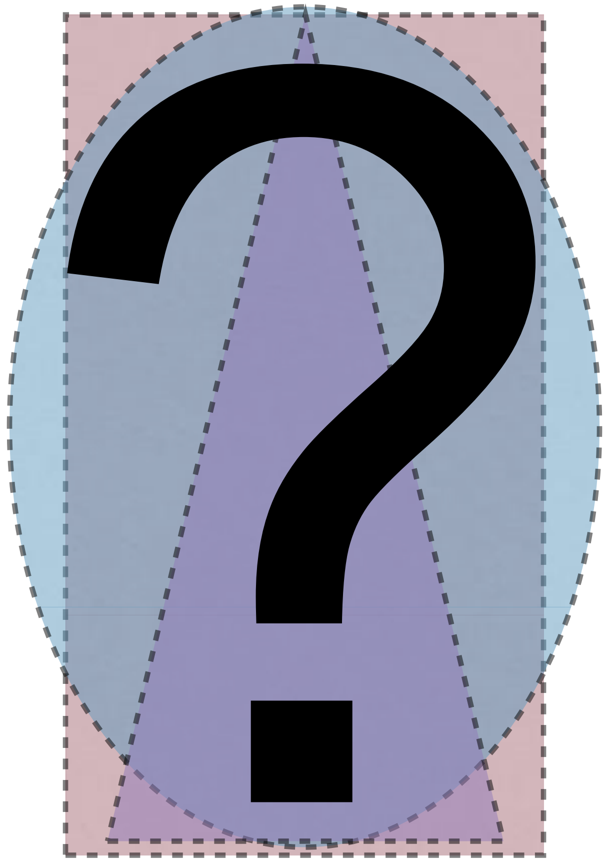


What are these?

Use generalisation and specialisation to create families of classes

What do you want to do with shape?

Do you care if they are ellipses, rectangles, triangles?



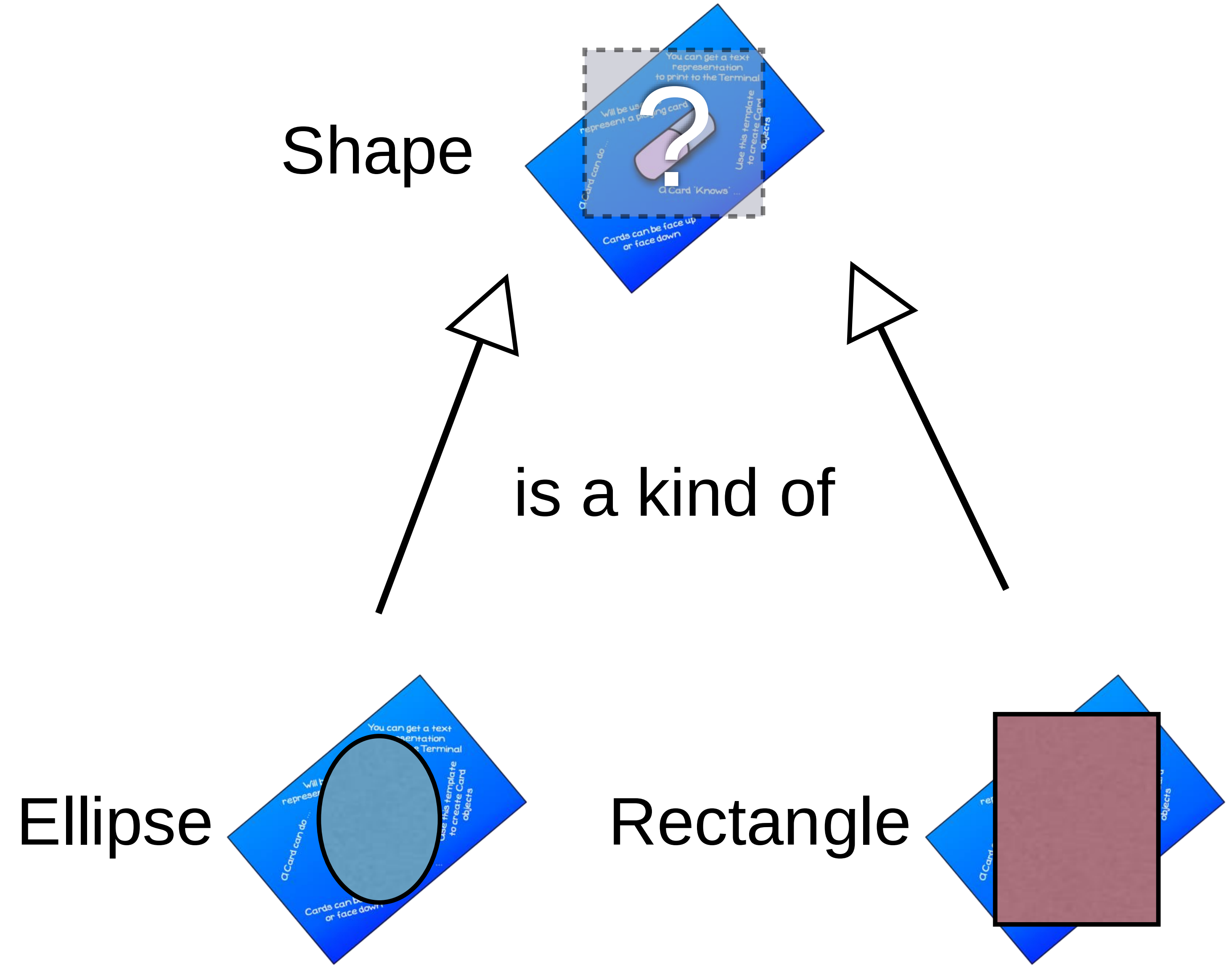
SWINBURNE UNIVERSITY OF TECHNOLOGY

Use inheritance to model generalisation and specialisation
in your OO code

Inherit attributes and behaviour from a **parent** class

SWINBURNE UNIVERSITY OF TECHNOLOGY

Use inheritance to model **is-a** kinds of relationships


Shape

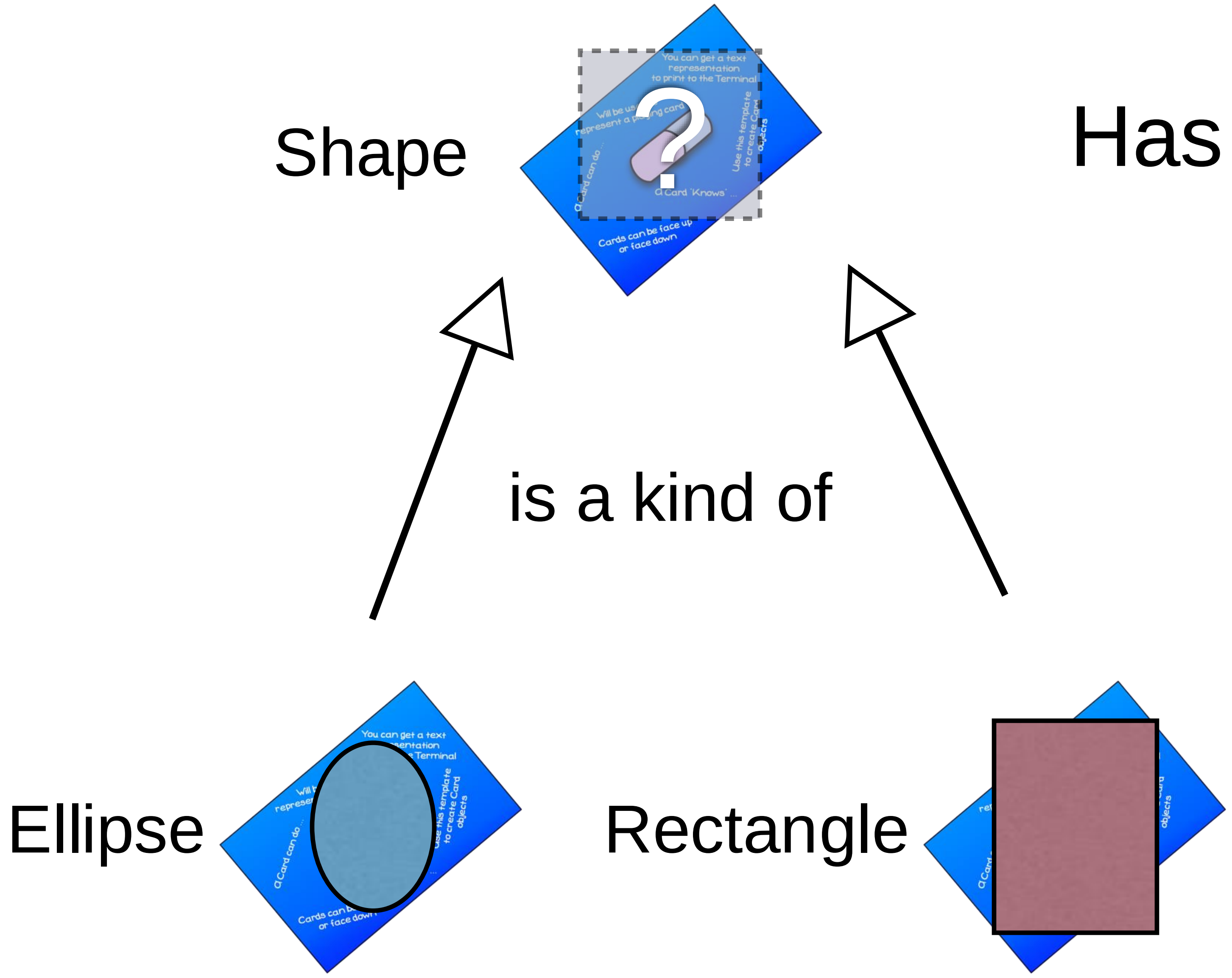
is a kind of

Ellipse

Rectangle

You can get a text representation to print to the Terminal
Will be used to represent a playing card
Use this template to create Card objects
a Card can do
a Card 'Knows'
Cards can be face up or face down

The child class **inherits** all of the features of the parent...



Shape

is a kind of

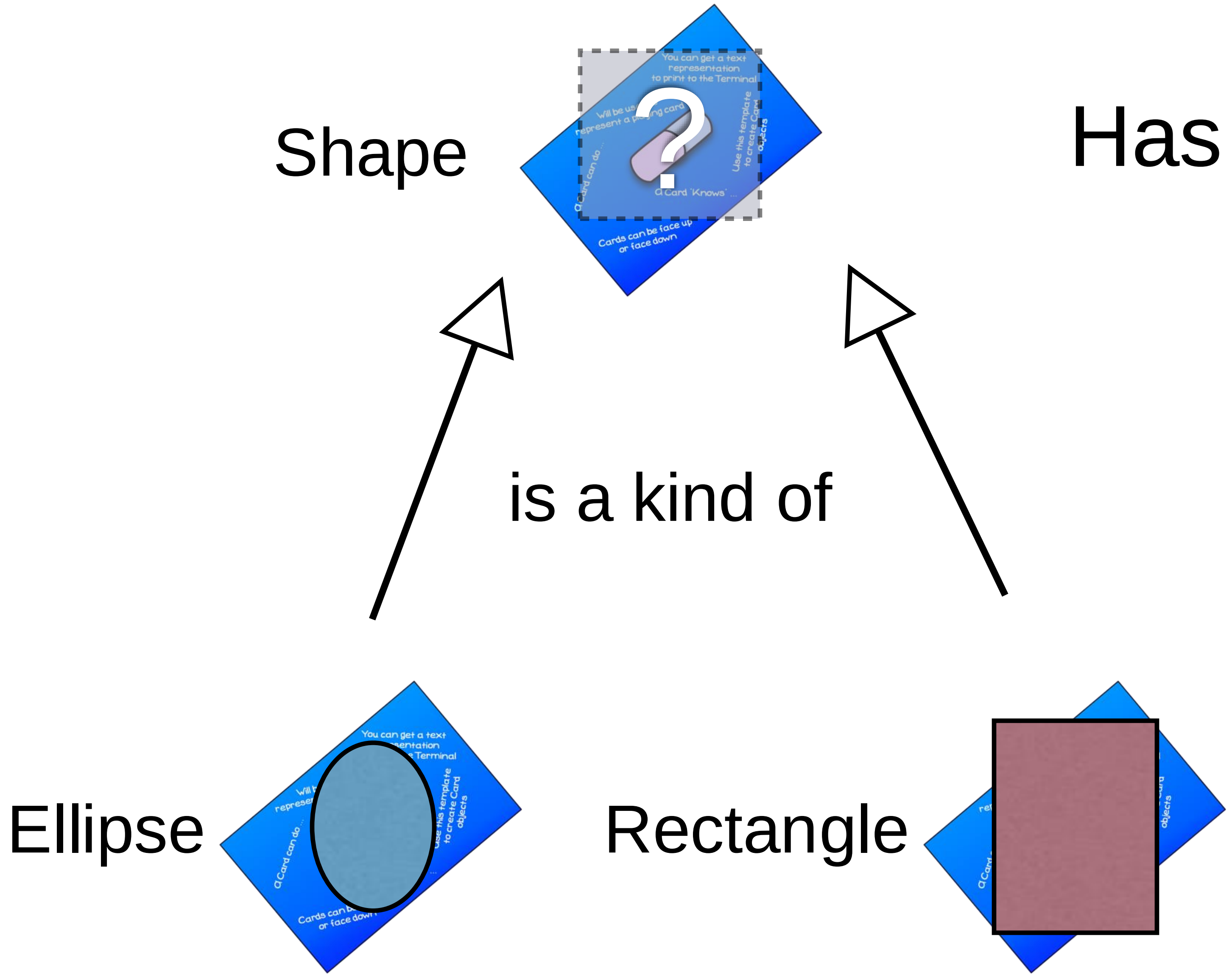
Ellipse

Rectangle

Has a position, size, and can be drawn.

inherits the position, size,
and can be drawn.

Change how inherited methods behave in the child class (**overriding** the parent)

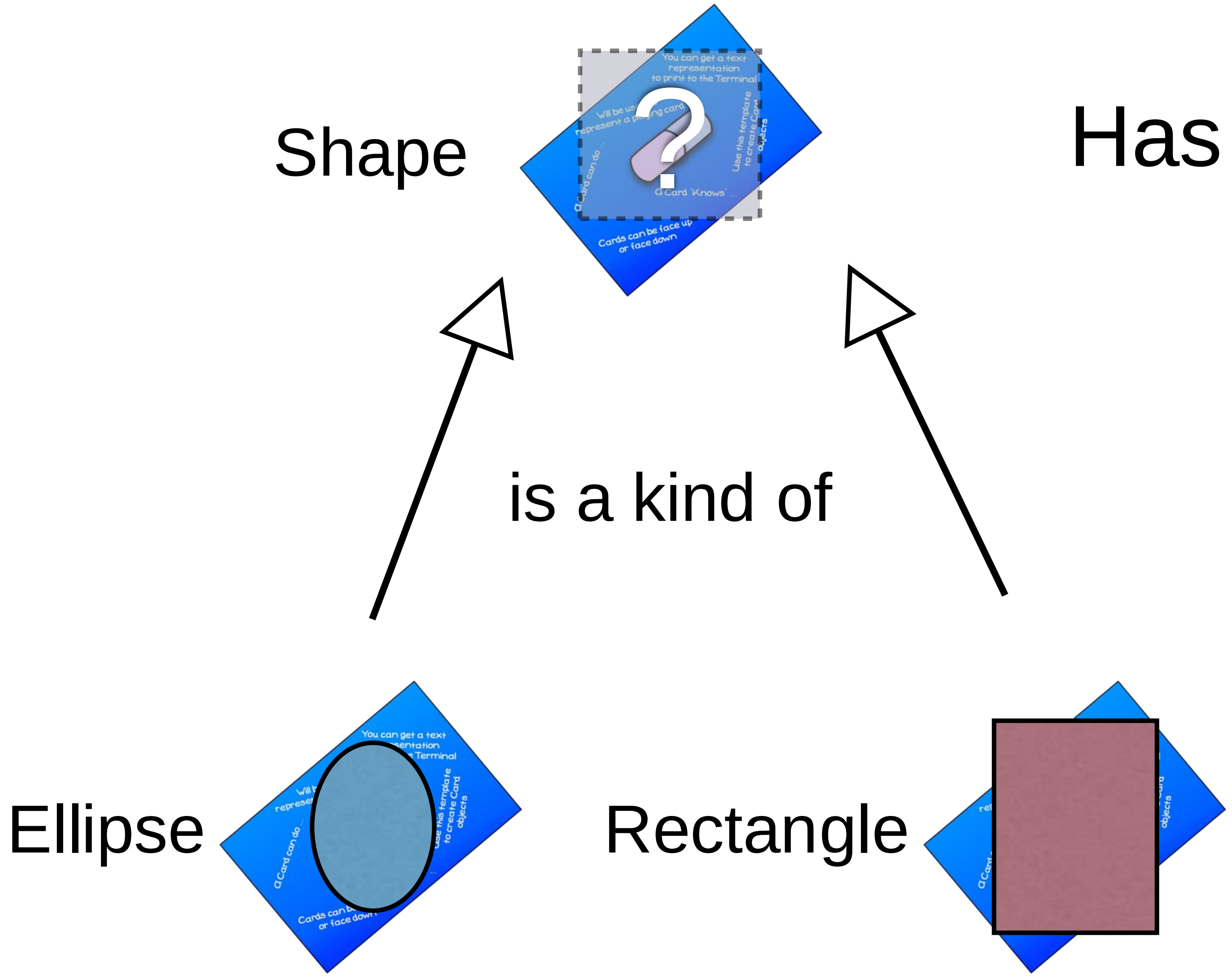


Has a position, size, and **can be drawn.**

Rectangle = Draws a Rectangle

Ellipse = Draws an Ellipse

Add additional features in the child class



Shape

is a kind of

Ellipse

Rectangle

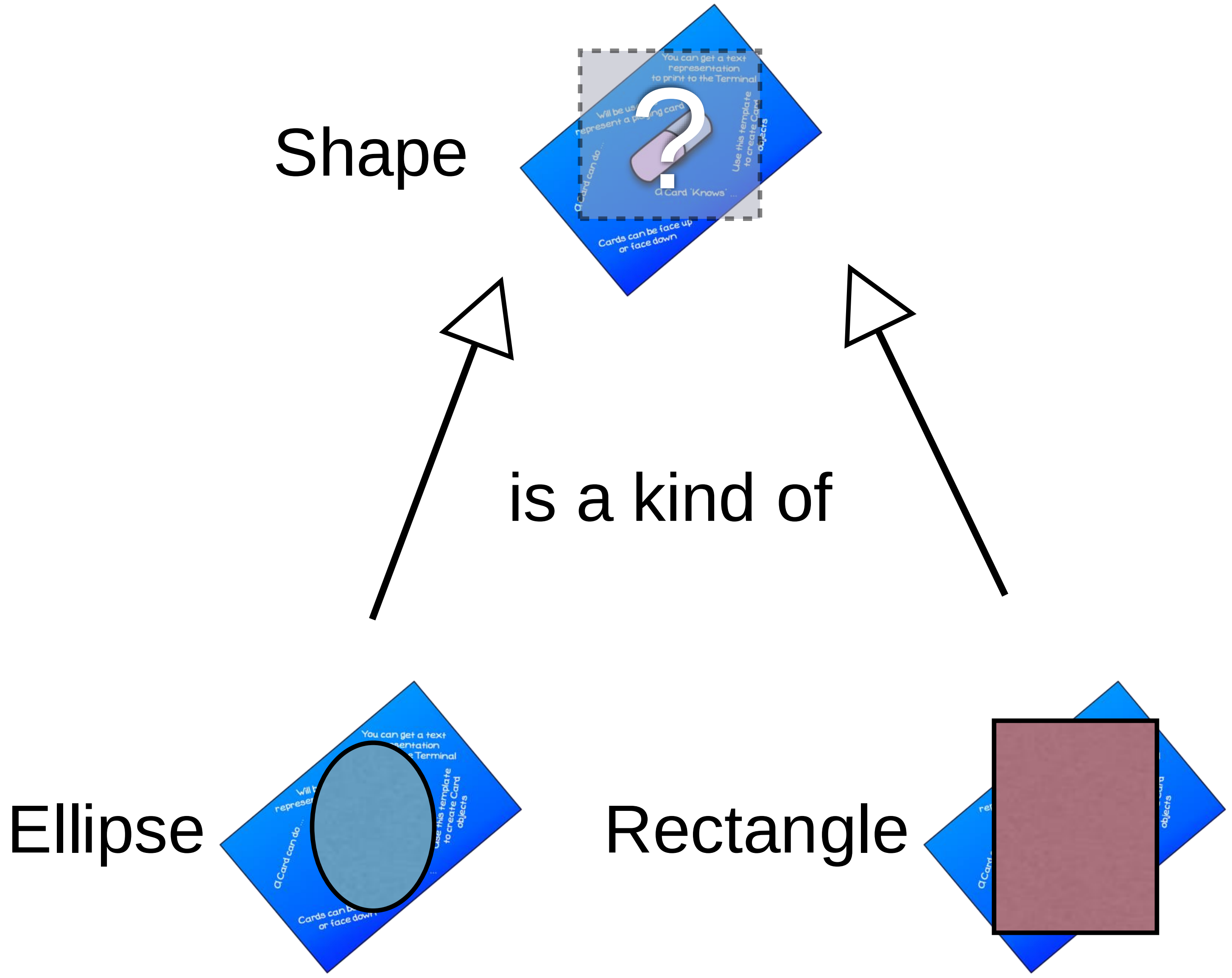
Has a position, size, and **can be drawn.**

Rectangle = MakeSquare()

Ellipse = MakeCircular()

SWINBURNE UNIVERSITY OF TECHNOLOGY

The child class can see public and protected members of the parent



Access levels

public: anyone

protected: only derived classes

private: nobody else

Inheritance declared by **derived classes**

C++

class Rectangle : public Shape

C#

class Rectangle : Shape

Java

class Rectangle extends Shape

Objective-C

@interface Rectangle : Shape

Take away message

- Inheritance is a huge core concept of OOP, helping you create object-oriented programs
- Abstraction is much more than just classification
- Inheritance helps us with abstraction, creating layers of generalisation/specialisation
- Inheritance helps bring flexibility, extensibility, and adaptability